Attorney's Docket No.: 12221-007001

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant	: Thomas Michael Gil et al.	Art Unit	: 2152
Serial No.	: 09/931,223	Examiner	: Nguyen, Trong Nhan P.
Filed	: August 16, 2001		
Title	: STATISTICS COLLECTION FOR NETWORK TRAFFIC		

MAIL STOP APPEAL BRIEF – PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF ON BEHALF OF THOMAS MICHAEL GIL ET AL.

The Appeal Brief fee of **$250** is enclosed. Please apply any other charges or credits to Deposit Account No. 06-1050.

03/10/2006 YPOLITE1 00000115 09931223
02 FC:2402 250.00 OP

CERTIFICATE OF MAILING BY FIRST CLASS MAIL

I hereby certify under 37 CFR §1.8(a) that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

March 7, 2006
Date of Deposit

Marie Collins
Signature

Marie Collins
Typed or Printed Name of Person Signing Certificate

(i.) Real Party In Interest

The real party in interest in the above application is Mazu Networks, Inc.

(ii.) Related Appeals and Interferences

The appellant is not aware of any appeals or interferences related to the above-identified patent application.

(iii.) Status of Claims

This is an appeal from the decision of the Primary Examiner in an Office Action dated October 20, 2005, rejecting claims 1-21 and 50-77, all of the claims in the application. Claims 22-49 were canceled. The claims have been twice rejected. Claims 1-21 and 50-77 are the subject of this appeal.

(iv.) Status of Amendments

All amendments have been entered. Appellant filed herewith a Notice of Appeal on **December 14, 2005**

(v.) Summary of Claimed Subject Matter

Background

This invention relates to techniques to thwart network-related denial of service attacks. [Specification page 1, lines 3-5]

In denial of service attacks, an attacker sends a large volume of malicious traffic to a victim preventing the victim from responding to legitimate traffic. [Specification page 1, lines 6-16].

Appellant's Invention

Claim 1

One aspect of Appellant's invention is set out in claim 1 as a machine implemented method of monitoring traffic flow in a monitoring device disposed to receive network traffic

packets. "Referring to FIG. 6, a monitoring process 32 is shown. The monitoring process 32 can be deployed on data collectors 28 as well as gateways 26." [Appellant's specification Page 13, lines 24-27].

Inventive features of claim 1 include producing statistics corresponding to a parameter of traffic flow to trace the source of an attack. "Referring to FIG. 4, the data collector 26 performs 40 a sampling and statistic collection process 40. The data collector samples 42 one (1) packet in every (n) packets and has counters to collect statistics about every packet." [Appellant's specification Page 9, lines 11-14]. "The gateways 26 and data collectors have monitoring process 32 used to measure some parameter of traffic flow. One goal of the gateways 26 and data collectors 28 is to measure some parameter of network traffic. This information collected by the gateways 26 and data collectors is used to trace the source of an attack." [Appellant's specification Page 14, lines 5-10].

Inventive features of claim 1 include mapping the traffic flow into a plurality of buckets by applying a hash function "f(h)" to the parameter of the traffic flow to output an integer corresponding to one of the buckets. "The algorithm will use some hash function "f(h)", which takes the packet and outputs an integer that corresponds to one of the buckets "B_1 - B_N."" [Appellant's specification Page 14, lines 18-21].

Inventive features of claim 1 include accumulating statistics from the packets and comparing the number of buckets to a threshold. "Statistics from the packets start accumulating in the buckets "B_1 - B_N". The buckets "B_1 - B_N" are configured with threshold values "Th." As the contents of the buckets B_1 - B_N reach the configured thresholds values "Th", (e.g., compare values of packet count or packet rate to threshold), the monitoring process 32 deems that event to be of significance." [Appellant's specification Page 14, lines 21-25].

Inventive features of claim 1 include determining whether the number of buckets should be divided into more buckets or combined into fewer buckets based on comparing the number of buckets to the threshold. "As the gateway 26 or data collector 28 approaches a bucket threshold "Th", the gateway 26 or data collector 28 have the ability to take several buckets $B_1 - B_3$ and divide them in more buckets $B_1 - B_4$ or combine them into fewer bucket $B_1 - B_2$." [Appellant's specification Page 15, lines 18-22].

Claim 14

Claim 14 claims another aspect of the invention. Claim 14 is a computer program product residing on a computer readable for monitoring network traffic flow in a network. [Appellant's specification Page 2, lines 8-12]. The gateway 26 and data collector 26 are typically software programs that are executed on devices such as computers, routers, or switches. [Appellant's specification Page 9, lines 6-8].

Inventive features of claim 14 include instructions to map traffic flow into a plurality of buckets by applying a hash function "f(h)" to a parameter of the traffic flow to output an integer corresponding to one of the buckets. This feature is supported as the analogous feature of claim 1.

Inventive features of claim 14 include instructions to accumulate statistics from the packets and compare the accumulated statistic values from the buckets to configured threshold values corresponding to the number of buckets to determine that an event is of significance. This feature is supported as the analogous feature of claim 1.

Inventive features of claim 14 include instructions to adjust the number of buckets as the number of buckets approaches a second threshold. This feature is supported as the analogous feature of claim 1.

Claim 21

Another aspect of the invention is covered by claim 21. Claim 21 is directed to a data collector to collect statistical information about network flows. "Referring to FIG. 4, the data collector 26 performs 40 a sampling and statistic collection process 40. The data collector samples 42 one (1) packet in every (n) packets and has counters to collect statistics about every packet." [Appellant's specification Page 9, lines 11-14].

Inventive features of claim 21 include a computer readable medium and a computing device that executes a computer program product stored on the computer readable medium. "The gateway 26 and data collector 26 are typically software programs that are executed on devices such as computers, routers, or switches." [Appellant's specification Page 9, lines 6-8].

Inventive features of claim 21 include instructions to map traffic flow into a plurality of buckets by applying a hash function "f(h)" to the parameter of the traffic flow to output an

integer corresponding to one of the buckets. This feature is supported as the analogous feature of claim 1.

Inventive features of claim 21 include instructions to accumulate statistics from the packets and compare the accumulated statistic values from the buckets to configured threshold values corresponding to the number of buckets to determine that an event is of significance. adjust the number of buckets as the number of buckets approaches a second threshold. This feature is supported as the analogous feature of claim 1.

Claim 63

Claim 63 is directed to a method of monitoring traffic flow in a monitor device disposed to receive network traffic packets. This feature is supported as the analogous feature of claim 1.

Inventive features of claim 63 include producing statistics corresponding to a parameter of traffic flow to trace the source of an attack. This feature is supported as the analogous feature of claim 1.

Inventive features of claim 63 include mapping the traffic flow into a plurality of buckets. This feature is supported as the analogous feature of claim 1.

Inventive features of claim 63 include varying the number of buckets according to the amount of traffic and number of flows according to down traffic flow into different buckets and examining statistics accumulated for a parameter and a corresponding threshold in the bucket. "As the gateway 26 or data collector 28 approaches a bucket threshold "Th", the gateway 26 or data collector 28 have the ability to take several buckets $B_1 - B_3$ and divide them in more buckets $B_1 - B_4$ or combine them into fewer bucket $B_1 - B_2$.

The function of the variable number of buckets is to dynamically adjust the monitoring process to the amount of traffic and number of flows, so that the monitoring device (e.g., gateway 26 or data collector 28) is not vulnerable to DoS attacks against its own resources. The variable number of buckets also efficiently identifies the source(s) of attack by breaking down traffic into different categories (buckets) and looking at the appropriate parameters and thresholds in each bucket." [Appellant's specification Page 15, lines 18-31].

Claim 70

Claim 70 is directed to a computer program product residing on a computer readable medium for monitoring traffic flow in a monitor device disposed to receive network traffic packets. This feature is supported as the analogous feature of claim 1.

Inventive features of claim 70 include instructions to produce statistics corresponding to a parameter of traffic flow to trace the source of an attack. This feature is supported as the analogous feature of claim 1

Inventive features of claim 70 include instructions to map the traffic flow into a plurality of buckets. This feature is supported as the analogous feature of claim 1.

Inventive features of claim 70 include instructions to vary the number of buckets according to the amount of traffic and number of flows according to down traffic flow into different buckets and examining statistics accumulated for a parameter and a corresponding threshold in the bucket. This feature is supported as the analogous feature of claim 63.

(vi.) The Ground of Rejection to be Reviewed on Appeal

Claims 1-21 and 50-77 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Belissent (USPN 6789203) further in view of Vaidya (USPN 6,279,113).

(vii.) Argument

Obviousness

"It is well established that the burden is on the PTO to establish a prima facie showing of obviousness, *In re Fritsch*, 972 F.2d. 1260, 23 U.S.P.Q.2d 1780 (C.C.P.A., 1972)."

"It is well established that there must be some logical reason apparent from the evidence or record to justify combination or modification of references. *In re Regal*, 526 F.2d 1399 188, U.S.P.Q.2d 136 (C.C.P.A. 1975). In addition, even if all of the elements of claims are disclosed in various prior art references, the claimed invention taken as a whole cannot be said to be obvious without some reason given in the prior art why one of ordinary skill in the art would have been prompted to combine the teachings of the references to arrive at the claimed invention. Id. Even if the cited references show the various elements suggested by the Examiner in order to

support a conclusion that it would have been obvious to combine the cited references, the references must either expressly or impliedly suggest the claimed combination or the Examiner must present a convincing line of reasoning as to why one skilled in the art would have found the claimed invention obvious in light of the teachings of the references. *Ex Parte Clapp*, 227 U.S.P.Q.2d 972, 973 (Board. Pat. App. & Inf. 985)."

"The mere fact that the prior art could be so modified would not have made the modification obvious unless the prior art suggested the desirability of the modification." *In re Gordon*, 221 U.S.P.Q. 1125, 1127 (Fed. Cir. 1984).

> Although the Commissioner suggests that [the structure in the primary prior art reference] could readily be modified to form the [claimed] structure, "[t]he mere fact that the prior art could be so modified would not have made the modification obvious unless the prior art suggested the desirability of the modification." *In re Laskowski*, 10 U.S.P.Q. 2d 1397, 1398 (Fed. Cir. 1989).

"The claimed invention must be considered as a whole, and the question is whether there is something in the prior art as a whole to suggest the desirability, and thus the obviousness, of making the combination." *Lindemann Maschinenfabrik GMBH v. American Hoist & Derrick*, 221 U.S.P.Q. 481, 488 (Fed. Cir. 1984).

> Obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion supporting the combination. Under Section 103, teachings of references can be combined only if there is some suggestion or incentive to do so. *ACS Hospital Systems, Inc. v. Montefiore Hospital*, 221 U.S.P.Q. 929, 933 (Fed. Cir. 1984) (emphasis in original, footnotes omitted).

"The critical inquiry is whether 'there is something in the prior art as a whole to suggest the desirability, and thus the obviousness, of making the combination.'" *Fromson v. Advance Offset Plate, Inc.*, 225 U.S.P.Q. 26, 31 (Fed. Cir. 1985).

Claims 1-21 and 50-77 are allowable over Belissent (USPN 6,789,203) and Vaidya (USPN 6,279,113).

Claims 1, 2, 6, 7, and 12.

For the purposes of this appeal only claims 1, 2, 6, 7, and 12 stand or fall together. Claim 1 is representative of this group of claims.

Claim 1 calls for a machine implemented method of monitoring traffic flow in a monitoring device disposed to receive network traffic packets. Claim 1 is distinct over the cited references, since the references whether taken separately or in combination fail to suggest producing statistics corresponding to a parameter of traffic flow to trace the source of an attack and mapping the traffic flow into a plurality of buckets.

The examiner contends with respect to these features of Appellant's claim 1 that:

> **As to claim 1, Belissent teaches ...**
> **producing statistics corresponding to a parameter of traffic flow to trace the source of an attack, (abstract; col 2, lines 55-65; the client's request rate is measured) with producing further comprising:**
> **accumulating statistics from the packets (abstract; col 2, lines 55-65; the client's request rate is measured)**
> **and comparing the number of buckets to a threshold (col 3, lines 6-36);**

In the cited passages from Belissent, the reference teaches: "a DoS defense module determines a connection request rate for a particular client. The client is blocked if the connection request rate is determined to be above a first pre-determined threshold. If, however, the connection request rate is below the first threshold but above a second threshold, then the client's connection request rate is slowed, or throttled, down to a rate consistent with a connection delay interval that's is based upon a throttling factor." [Belissent, Abstract].

The examiner also contends that Belissent, at (col 3, lines 6-36) teaches comparing the number of buckets to a threshold. Belissent describes:

> **The apparatus also includes a processor unit coupled to the interval m connection request count buffer arrainged (sic) to determine if the interval m connection request count is greater than a rejection threshold associated with the requesting client and a request throttler unit coupled to the processor unit arrainged (sic) to reject the connection request when it is determined that the interval m connection request count is greater than the rejection threshold, and wait**

an interval m wait time when it is determined that the interval m connection request count is not greater than the rejection threshold before the request is accepted by the server computer.

In another embodiment of the invention, computer readable media including computer program code for preventing a denial of service (DoS) attack by a requesting client on a server computer is disclosed. The computer readable medium includes computer program code for receiving a connection request at a time t_n in a throttling interval m, computer program code for incrementing an interval m connection request count if the time t_n is not at a beginning of the throttling interval m, and computer program code for determining if the interval m connection request count is greater than a rejection threshold associated with the requesting client. The computer readable medium also includes computer program code for rejecting the connection request if it is determined that the interval m connection request count is greater than the rejection threshold, computer program code for waiting an interval m wait time if it is determined that the interval m connection request count is not greater than the rejection threshold, and computer program code for accepting the request by the server computer. Belissent, (col 3, lines 6-36)

Appellant contends that connection request rates, as disclosed in Belissent, do not suggest producing statistics corresponding to a parameter of traffic flow to trace the source of an attack nor mapping the traffic flow into a plurality of buckets.

Nevertheless, assuming *arguendo* that the examiner is correct, Belissent in the passage disclosed above and well as elsewhere fails to suggest the feature of comparing the number of buckets to a threshold. One of ordinary skill in the art would not interpret the passage cited by the examiner, as disclosing or suggesting this feature. Belissent has a request count buffer arranged to determine if over an interval, the connection request count is greater than a rejection threshold associated with the requesting client. Belissent, teaches to reject the connection request when the connection request count is greater than a rejection threshold, or wait when the connection request count is not greater than the rejection threshold. Belissent, also has an embodiment in which Belissent, throttles the requests.

However, Belissent, does not suggest comparing the number of buckets to a threshold. Rather, Belissent compares the count, i.e., the value in the counter to the threshold, not the number of buckets. However, comparing the count in a counter to a threshold is not the equivalent of nor would suggest comparing the number of buckets to a threshold, as claimed. That is, Belissent stores a count of connection requests for a particular client but does not suggest to compare the number of counters to a threshold of the number of counters.

The Examiner's further contends that:

As to claim 1, Belissent teaches ...

and determining whether the number of buckets should be divided into more buckets or combined into fewer buckets based on comparing the number of buckets to the threshold (col 2, line 49 to col 3, line 36; the rate of connections is compared to thresholds and appropriate action is taken).

However, Bellisent (sic) does not explicitly indicate mapping the traffic flow into a plurality of buckets by applying a hash function "f(h)" to the parameter of the traffic flow to output an integer corresponding to one of the buckets.

Vaidya teaches mapping the traffic flow to a memory space by applying a hash function (col 3, lines 27-48; col 9, lines 3-20). It would have been obvious to one of ordinary skill in the art at the time of the invention to incorporate the teachings of Vaidya into those of Belissent in order to make the system more organized. The organization of the system using a hash function is known in the art. A specific hash function is used because its backward computation is difficult as well as its tendency to be collision free.

Belissent fails to suggest comparing the number of buckets to a threshold. Belissent also fails to suggest: "determining whether the number of buckets should be divided into more buckets or combined into fewer buckets based on comparing the number of buckets to the threshold."

The examiner contends that determining whether the number of buckets feature is taught by Belissent at col. 2, line 49 to col. 3, line 36. However, no such teaching is found at the cited passages or elsewhere in Belissent. Rather, at that passage Belissent merely discloses an algorithm on when to update connection request counts and whether or not to accept connection requests. Nothing in Belissent teaches to vary the number of buckets, e.g., "counters" disclosed in Belissent by comparison of the number of counters to a threshold, or to determine whether the number of "counters" should be divided into more counters or combined into fewer counters based on comparing the number of counters to the threshold.

It would not be relevant to Belissent to divide counters according to any threshold because, Belissent uses the counters to track connection requests base on client IP address, for the purpose of determining whether or not to grant access. That is why Belissent merely teaches to compare the counts, i.e., the values in the counters to a threshold of connection request rates to grant or deny or throttle connection requests for the connection. Hence, Belissent does not teach or suggest the claimed feature of determining whether the number of buckets should be divided into more buckets or combined into fewer buckets.

The examiner admits that Belissent fails to suggest "... applying a hash function to the parameter of the traffic flow to output an integer corresponding to one of the buckets." and therefore relies on Vaidya for this feature.

Vaidya fails to suggest: "mapping the traffic flow into a plurality of buckets by applying a hash function "f(h)" to the parameter of the traffic flow." While Vaidya indeed mentions the use of a "hash index" to access a cache, Vaidya fails to suggest a hash of a parameter of traffic flow or to use the hash to map traffic into the plurality of buckets. Vaidya uses the hash index to search a cache for matching session entries in the cache, a common technique employed to distribute entries in a cache memory.

One of ordinary skill in the art would not be motivated to apply Vaidya to Belissent as a technique to map traffic flow. This lack of motivation follows because Belissent teaches: "The throttler unit 208 includes a connection request monitor 210 arrainged (sic) to monitor the number of connection requests received by a particular requesting client based upon the requesting clients unique IP address." Since, Belissent tracks connection requests on a per client basis to determine whether to grant or deny a request, modification of Belissent by applying a hash, as the examiner contends would be superfluous at best or inoperative at worst, since Belissent would not map connection requests to counters, because the counters are assigned based on the requesting client's IP addresses.

Accordingly, Vaidya adds no further teachings to cure the deficiencies in Belissent and therefore Belissent taken together or separately with Vaidya fails to suggest claim 1.

Claim 3

Claim 3 further limits claim 1 by reciting that as the number of buckets changes, the buckets have values derived from the buckets prior to the change. Claim 1 recites the feature of determining whether the number of buckets should be divided into more buckets or combined into fewer buckets based on comparing the number of buckets to the threshold. As discussed above, the combination of references do not suggest that the number of buckets changes based on a comparison to a threshold.

Appellant discusses in the specification (page 14, line 27) that:

> The monitoring process 32 takes that bucket, e.g., B_i and divides that bucket Bi into some other number M of new buckets $B_{i1} - B_{iM}$. Each of the new buckets $B_{i1} - B_{iM}$ contains values appropriately derived from the original bucket B_i.

It would serve no purpose to combine Belissent with Vaidya to derive contents for the disclosed counters, since neither Belissent nor Vaidya teach to divide the counters in the first instance, and any combination of these references would not suggest to derived contents for the new buckets.

Appellant contends that the intent of Belissent is to track connection requests and to protect a system against an attack. Belissent accomplishes this by determining a connection request rate for a given IP address. However, dividing counters into more counters and deriving values would seem to have little benefit to Belissent, since it would complicate the action of determining connection rates and whether or not to grant access, without any purported advantage offered by the examiner.

Claim 4

Appellant also describes (page 14, line 31) that:

> Also, the hash function is extended to map to N+M-1 "h→N+M-1" values, rather than the original N values. [Appellant's specification page 14, line 31 to page 15, line 2]

Belissent and Vaidya fail to disclose the concept of dividing buckets and hence fail to suggest that the hash function adapts to map to the new number of buckets, as the new number of buckets changes, as claimed in claim 4. The examiner contends that Vaidya teaches this feature at (col. 9, lines 3-45).

Vaidya at the cited passage teaches operation of a session cache with a hash index and does mention options in the manner that the hash index is formed. However, Vaidya is devoid of any suggestion of a hash function that adapts to map to the new number of buckets, as the new number of buckets changes. Vaidya does not adapt a hash function for any purpose and in particular based on changes in the number of buckets.

Claim 5

Claim 5 further limits claim 1 by comparing the value accumulated in the bucket to a threshold that depends on the number of buckets.

The examiner contends that Belissent teaches this feature at col. 2, line 49 to col. 3, line 36. Appellant disagrees.

Rather, Belissent teaches to maintain a connection request count. If a connection request rate based on the count is determined to be greater than a rejection threshold associated with the requesting client, then Belissent teaches to reject the connection request. Belissent does not teach a threshold dependent on the number of buckets. Rather, Belissent teaches to vary the acceptance rate of connection requests according to the connection request count compared to the rejection threshold. Belissent does not teach that the rejection threshold varies according to any parameter and especially according to the number of connection requests.

Belissent teaches at Col. 4 lines 26-49 that: "time is divided into intervals (one such interval is called a throttling interval) in which the number of connections per client IP address is recorded." Belissent also teaches that: "If a particular client's connection request rate is greater than a rejection threshold associated with that client, the IP throttler will refuse any new connections from the client until the beginning of the next throttling interval."

Belissent makes provision for "a slowdown threshold" that "provides the maximum number of connections per time interval (such as a throttling interval) from a particular IP address that the server is willing to accept without slowing down new incoming connections." However, this slowdown threshold again does not appear to vary with the number of connections and if exceeded merely applies "a wait time ... to delay the incoming connection request stream." Belissent describes that: "the wait time is related to the number of connection requests (hits) above the slowdown threshold as referred to as a slowdown rate. For example, the slowdown rate represents how many hourly connections in excess of the slowdown threshold will cause one second of wait time." Nowhere does Belissent teach comparing the value accumulated in the bucket to a threshold that depends on the number of buckets.

Claim 8

Claim further limits claim 1 and requires that the hash function changes periodically in a randomly, secret manner so that packets are reassigned to different buckets.

Claim 8 is neither described nor suggested by Belissent and Vaidya. The examiner admits that Belissent fails to describe a hash function and can only produce Vaidya to teach the hash function. However, Vaidya teaches a producing of a "hash index" to access a cache, is actually silent on a hash function, and Vaidya fails to suggest to apply a hash function to a parameter of traffic flow or to use a hash function to map traffic into a plurality of buckets.

The examiner contends that the use of a hash function that changes, in a random secret manner is found in Vaidya Col. 9, lines 3-45. Appellant is unable to find any support for this contention. Appellant contends that it would not be apparent why Belissent and Vaidya would use a secret hash function. Clearly, Vaidya would not benefit from using of a randomly changing secret hash function to produce a hash index to search a cache for matching session entries in the cache, and neither Belissent nor Vaidya would appear to benefit from such a resulting reassignment of packets to different buckets, since again Belissent seeks to track connection requests associated with the requesting client. Assigning to different buckets would defeat the purpose of Belissent to track connections on a requesting client IP address basis.

Claim 9

Claim 9 depends on claim 1 and calls for the variable number of buckets (supported by dividing buckets into more or fewer buckets, as recited in claim 1) that dynamically adjusts the amount of traffic and number of flows monitored, so that the monitoring device is not vulnerable to a denial of service attack against its own resources. Belissent fails to teach adjusting the number of buckets to protect against a denial of service attack. Rather Belissent teaches to throttle connection request, or deny connection requests to protect against a denial of service attack.

Claim 10

Claim 10 depends from claim 1 and recites that the variable number of buckets efficiently identifies the source or sources of attack by breaking down traffic into different buckets and

examining statistics accumulated for a parameter and a corresponding threshold in each bucket. Belissent fails to break traffic down in different buckets, rather Belissent merely monitors connection requests. Belissent also fails to suggest examining statistics accumulated for a parameter and a corresponding threshold in each bucket. Belissent teaches a rejection threshold associated with that client and refuses new connections from the client until the beginning of the next throttling interval if the threshold is exceeded. Belissent also teaches "a slowdown threshold" that is maximum number of connections per time interval from a particular IP address that the server is willing to accept without slowing down new incoming connections. These teachings are not a variable number of buckets that efficiently identifies the source or sources of attack by breaking down traffic into different buckets and examining statistics accumulated for a parameter and a corresponding threshold in each bucket.

Claim 11

Claim 11 further limits the method of claim 1 featuring that the traffic is monitored at multiple levels of granularity, from aggregate to individual flows. Belissent only monitors connection requests in counter according to client IP address, a single level, and hence fails to suggest monitoring a multiple levels of granularity ... .

Claim 13

Claim 13 depends on claim 1 and further recites that the threshold is a first threshold and adds a feature of comparing accumulated statistic values from the buckets to second threshold values to determine that an event is of significance.

While Belissent teaches "a slowdown threshold," as the maximum number of connections per time interval from a particular IP address that the server is willing to accept without slowing down new incoming connections, this threshold is not used as a flag or condition to indicate an event of significance.

Claims 14, 18, 19, 21, 53, 54, 57, 60, 61, 62, 77

For the purposes of this appeal only claims 14, 18, 19, 21, 53, 54, 57, 60, 61, 62, 77 stand or fall together. Claim 14 is representative of this group of claims.

Claim 14 recites instructions to map, accumulate, compare and adjust, in an analogous manner as the corresponding features of claim 1. Claim 14 does not recite the feature of producing statistics, as in claim 1.

Claim 14 distinguishes over the art since the cited references whether taken separately or in combination fail to suggest map the traffic flow into a plurality of buckets by applying a hash function to the parameter of the traffic flow to output an integer corresponding to one of the buckets." The examiner admits that Belissent fails to suggest apply a hash function ... and relies on Vaidya for this feature.

Vaidya fails to suggest: "mapping the traffic flow into a plurality of buckets by applying a hash function "f(h)" to the parameter of the traffic flow," as argued above. The "hash index" describe in Vaidya is to access a cache. Vaidya does not apply the hash to a parameter of traffic flow or to use the hash to map traffic into the plurality of buckets.

One of ordinary skill in the art would not be motivated to apply Vaidya to Belissent as a technique to map traffic flow. This lack of motivation follows because Belissent tracks connection requests on a per client basis to determine whether to grant or deny a request. The purported modification of Belissent to apply a hash, as the examiner contends would be superfluous at best or inoperative at worst because the counters are pre-assigned based upon the requesting clients unique IP addresses.

Belissent and Vaidya taken together or separately fails to suggest instructions to adjust the number of buckets as the number of buckets approaches a second threshold. Belissent, teaches to throttle connection rates, but does not teach to adjust the number of buckets.

Claims 15 and 50

For the purposes of this appeal only claims 15 and 50 stand or fall together. Claim 15 is representative of this group of claims.

Claim 15 requires that based on the second threshold, the buckets are divided into more buckets or combined into fewer buckets. The second threshold disclosed by Belissent teaches to throttle connection requests not to divide or combine buckets.

Claims 16 and 51

For the purposes of this appeal only claims 16 and 51 stand or fall together. Claim 16 is representative of this group of claims.

Claim 16 further limits claim 14 and recites instructions to divide the bucket into a different number of new buckets containing values derived from the original bucket. Belissent does not teach to divide buckets and because Belissent uses counters based on IP address of a requesting client, would not inherently suggest dividing a bucket (or counter) into a different number and derive values from the original bucket.

Claims 17 and 52

For the purposes of this appeal only claims 17 and 52 stand or fall together. Claim 16 is representative of this group of claims.

Claim 17 further limits claim 14 to require the hash function adapt to map to the new number of buckets as the new number of buckets changes. Belissent does not teach a hash as acknowledged by the examiner. The examiner contends that Vaidya teaches this feature at (col. 9, lines 3-45).

Vaidya at the cited passage teaches operation of a session cache with a hash index and does mention options in the manner that the hash index is formed. However, Vaidya is devoid of any suggestion of a hash function that adapts to map to the new number of buckets, as the new number of buckets changes. Vaidya does not adapt a hash function for any purpose and in particular based on changes in the number of buckets.

Claims 20 and 55

For the purposes of this appeal only claims 20 and 55 stand or fall together. Claim 20 is representative of this group of claims.

Claim 55 further limits claim 21 to a hash function that changes periodically in a randomly secret manner so that packets are reassigned to different buckets. Belissent as admitted by the examiner fails to suggest a hash function. Vaidya would have no use for a secret hash function since Vaidya merely uses a hash index to access a cache.

Claim 56

Claim 56 further limits claim 21 by comparing the value accumulated in the bucket to a threshold that depends on the number of buckets. This claim is allowable for analogous reasons given in claim 5, namely that Belissent teachings to maintain a connection request count does not suggest a threshold that depends on the number of buckets. Rather, Belissent teaches to vary the acceptance rate of connection requests according to the connection request count compared to the rejection threshold. Belissent does not teach that the rejection threshold varies according to any parameter and especially according to the number of connection requests.

Claim 58

Claim 58 further limits claim 21 reciting that the variable number of buckets dynamically adjusts the amount of traffic and number of flows monitored, so that the data collector is not vulnerable to a denial of service attack against its own resources. Belissent fails to suggest a data collector and rather teaches a mechanism that is executed on a server. Thus, while the data collector can be disposed to protect a server or a victim data center, the data collector itself could be subject to a denial of service attack against its own resources. Variable number of buckets dynamically prevents such an exploit.

Claim 59

Claim 59 further limits claim 21 to a data collector that uses the variable number of buckets to efficiently identify the source or sources of an attack by breaking down traffic into different buckets and examining statistics accumulated for a parameter and a corresponding threshold in each bucket. Belissent does not suggest this feature. True Belissent maintains counters on an IP basis that count connection requests, and if the IP address is not spoofed, Belissent may be capable of tracing the source of an attack. However, Belissent cannot use those counters to determine the source of an attack, if the IP addresses are spoofed. Indeed, Belissent does not use values of the counts in the counters to determine the sources of an attack, but rather the IP address corresponding to the counter. In contrast, Claim 59 requires that the statistics accumulated for a parameter and a corresponding threshold in each bucket are used to identify the source of an attack.

Claims 63, 66, 70 and 73

For the purposes of this appeal only claims 63, 66, 70 and 73 stand or fall together. Claim 63 is representative of this group of claims.

Claim 63 is directed to a method of monitoring traffic flow in a monitor device disposed to receive network traffic packets. Claim 63 includes the features of producing statistics corresponding to a parameter of traffic flow to trace the source of an attack. According to claim 63 producing includes mapping the traffic flow into a plurality of buckets and varying the number of buckets according to the amount of traffic and number of flows according to down[stream] traffic flow into different buckets and examining statistics accumulated for a parameter and a corresponding threshold in the bucket.

Belissent taken with Vaidya fails to suggest producing statistics corresponding to a parameter of traffic flow, Belissent merely counts the number of connection requests, but does not produce a statistic related to a parameter of traffic flow and does not produce statistics to trace the source of an attack. Belissent merely produces the count of connection requests from a particular client to determine whether to deny, grant or throttle connection requests.

Claims 64, 68, 71, and 75

For the purposes of this appeal only claims 64, 68, 71 and 75 stand or fall together. Claim 64 is representative of this group of claims.

Claim 64 further limits claim 63 and recites that: "varying varies the number of buckets so that the monitoring device is not vulnerable to DoS attacks against its own resources." This feature is not taught by any combination of Belissent and Vaidya. Belissent teaches a server that protects against DoS attacks by throttling connection requests. However, claim 64 specifically recites a monitor device to receive network traffic and the varying the number of buckets protects the monitor against a DoS attack on its own resources.

Claims 65 and 72

For the purposes of this appeal only claims 65 and 72 stand or fall together. Claim 65 is representative of this group of claims.

Claim 65 further limits claim 63 and recites that varying the number of buckets includes comparing the number of buckets to a threshold number of buckets and determining whether the number of buckets should be divided into more buckets or combined into fewer buckets based on comparing the number of buckets to the threshold and as the number of buckets changes, the buckets have values derived from the buckets prior to the change.

As discussed above, the combination of references do not suggest that the number of buckets changes based on a comparison to a threshold. It would serve no purpose to combine Belissent with Vaidya to derive contents for the disclosed counters, since neither Belissent nor Vaidya teach to divide the counters in the first instance, and any combination of these references would not suggest to derived contents for the new buckets.

Appellant contends that the intent of Belissent is to track connection requests and to protect a system against an attack. Belissent accomplishes this by determining a connection request rate for a given IP address. However, dividing counters into more counters and deriving values would seem to have little benefit to Belissent, since it would complicate the action of determining connection rates and whether or not to grant access. Since Belissent and Vaidya fail to disclose the concept of dividing buckets they inherently fail to suggest that the hash function adapts to map to the new number of buckets, as the new number of buckets changes. Vaidya is devoid of any suggestion of a hash function that adapts to map to the new number of buckets, as the new number of buckets changes. Vaidya does not adapt a hash function for any purpose and in particular based on changes in the number of buckets.

Claims 67 and 74

For the purposes of this appeal only claims 67 and 74 stand or fall together. Claim 67 is representative of this group of claims.

Claim 67 further limits claim 63 where comparing statistic values includes accumulating statistic values ... and comparing the values ... to thresholds that depend on the number of buckets. Belissent with Vaidya fails to suggest this feature. In Belissent the number of buckets is based on the number of client IP addresses that the server is tracking connection requests for. (See Col. 5, line 36 to Col. 6, line 18).

Claims 69 and 76

For the purposes of this appeal only claims 69 and 76 stand or fall together. Claim 69 is representative of this group of claims.

Claim 69 limits the method of claim 63 wherein the buckets are storage areas in a memory space of the monitor device and mapping the traffic flow into a plurality of buckets comprises applying a hash function "f(h)" to the parameter of the traffic flow to output an integer corresponding to one of the buckets.

The examiner admits that Belissent fails to suggest applying a hash. Appellant contends that Belissent fails to suggest "and mapping the traffic flow into a plurality of buckets comprises applying a hash function "f(h)" to the parameter of the traffic flow to output an integer corresponding to one of the buckets" and likewise Vaidya fails to suggest this feature. While Vaidya mentions a "hash index" to access a cache, Vaidya fails to suggest a hash of a parameter of traffic flow to output an integer corresponding to one of the buckets.

One of ordinary skill in the art would not be motivated to apply Vaidya to Belissent as a technique to map traffic flow since, Belissent tracks connection requests on a per client basis to determine whether to grant or deny a request and therefore modification of Belissent by applying a hash, as the examiner contends would be superfluous at best or inoperative at worst, since Belissent would not map connection requests to counters, because the counters are pre-assigned based upon the requesting clients unique IP addresses.

Conclusion

Appellant submits, therefore, that Claims 1-30 and 32 are allowable over the cited art. Therefore, the Examiner erred in rejecting Appellant's claims and should be reversed.

Respectfully submitted,

Date: 3/7/06

Denis G. Maloney
Reg. No. 29,670

Fish & Richardson P.C.
225 Franklin Street
Boston, MA 02110-2804
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

Appendix of Claims

1. A machine implemented method of monitoring traffic flow in a monitoring device disposed to receive network traffic packets comprises:

producing statistics corresponding to a parameter of traffic flow to trace the source of an attack, with producing further comprising:

mapping the traffic flow into a plurality of buckets by applying a hash function "f(h)" to the parameter of the traffic flow to output an integer corresponding to one of the buckets;

accumulating statistics from the packets; and

comparing the number of buckets to a threshold; and

determining whether the number of buckets should be divided into more buckets or combined into fewer buckets based on comparing the number of buckets to the threshold.

2. The method of claim 1 wherein the buckets are storage areas in a memory space of the monitor device.

3. The method of claim 1 wherein as the number of buckets changes, the buckets have values derived from the buckets prior to the change.

4. The method of claim 1 wherein the hash function adapts to map to the new number of buckets, as the new number of buckets changes.

5. The method of claim 1 wherein comparing statistic values comprises:

comparing the value accumulated in the bucket to a threshold that depends on the number of buckets.

6. The method of claim 1 wherein the parameter is the count of how many packets a data collector or gateway examines.

7. The method of claim 1 wherein as a value of a parameter for one bucket approaches a threshold, the monitoring device raises an alarm.

8. The method of claim 1 wherein the hash function changes periodically in a randomly secret manner so that packets are reassigned to different buckets.

9. The method of claim 1 wherein the variable number of buckets dynamically adjusts the amount of traffic and number of flows monitored, so that the monitoring device is not vulnerable to a denial of service attack against its own resources.

10. The method of claim 1 wherein the variable number of buckets efficiently identifies the source or sources of attack by breaking down traffic into different buckets and examining statistics accumulated for a parameter and a corresponding threshold in each bucket.

11. The method of claim 1 wherein the traffic is monitored at multiple levels of granularity, from aggregate to individual flows.

12. The method of claim 1 wherein the method is applied to monitoring of TCP packet ratios and repressor traffic.

13. The method of claim 1 wherein the threshold is a first threshold and the method further comprises:

comparing accumulated statistic values from the buckets to second threshold values to determine that an event is of significance.

14. A computer program product residing on a computer readable for monitoring network traffic flow in a network comprises instructions for causing a computer to:

map traffic flow into a plurality of buckets by applying a hash function "f(h)" to a parameter of the traffic flow to output an integer corresponding to one of the buckets;

accumulate statistics from the packets; and

compare the accumulated statistic values from the buckets to configured threshold values corresponding to the number of buckets to determine that an event is of significance; and

adjust the number of buckets as the number of buckets approaches a second threshold.

15. The computer program product of claim 14 wherein based on the second threshold, the buckets are divided into more buckets or combined into fewer buckets

16. The computer program product of claim 14 wherein instructions to monitor further comprise instructions to

divide the bucket into a different number of new buckets containing values derived from the original bucket.

17. The computer program product of claim 14 wherein the hash function adapts to map to the new number of buckets as the new number of buckets changes.

18. The computer program product of claim 14 wherein the parameter is the count of how many packets a data collector or gateway examines.

19. The computer program product of claim 14 wherein the buckets are storage areas in the memory space of the monitor device.

20. The computer program product of claim 14 wherein the hash function changes periodically in a randomly secret manner so that packets are reassigned to different buckets.

21. A data collector to collect statistical information about network flows comprises:

a computer readable medium;

a computing device that executes a computer program product stored on the computer readable medium comprising instructions to cause the computing device to:

map traffic flow into a plurality of buckets by applying a hash function "f(h)" to the parameter of the traffic flow to output an integer corresponding to one of the buckets;

accumulate statistics from the packets; and

compare the accumulated statistic values from the buckets to configured threshold values corresponding to the number of buckets to determine that an event is of significance; and

adjust the number of buckets as the number of buckets approaches a second threshold.

Claims 22-49 are canceled.

50. The data collector of claim 21 wherein based on the second threshold, the buckets are divided into more buckets or combined into fewer buckets

51. The data collector of claim 21 wherein instructions to monitor further comprise instructions to

divide the bucket into a different number of new buckets containing values derived from the original bucket.

52. The data collector of claim 21 wherein the hash function adapts to map to the new number of buckets as the new number of buckets changes.

53. The data collector of claim 21 wherein the parameter is the count of how many packets the data collector examines.

54. The data collector of claim 21 wherein the buckets are storage areas in the memory space of the monitor device.

55. The data collector of claim 21 wherein the hash function changes periodically in a randomly secret manner so that packets are reassigned to different buckets.

56. The data collector of claim 21 wherein instructions to compare statistic values comprises instructions to:

compare the value accumulated in the bucket to a threshold that depends on the number of buckets.

57. The data collector of claim 21 wherein as a value of a parameter for one bucket approaches a threshold, the monitoring device raises an alarm.

58. The data collector of claim 21 wherein the variable number of buckets dynamically adjusts the amount of traffic and number of flows monitored, so that the data collector is not vulnerable to a denial of service attack against its own resources.

59. The data collector of claim 21 wherein the variable number of buckets efficiently identifies the source or sources of attack by breaking down traffic into different buckets and examining statistics accumulated for a parameter and a corresponding threshold in each bucket.

60. The data collector of claim 21 wherein the traffic is monitored at multiple levels of granularity, from aggregate to individual flows.

61. The data collector of claim 21 wherein the traffic is applied to monitoring of TCP packet ratios and repressor traffic.

62. The data collector of claim 21 wherein the threshold is a first threshold and the computer program further comprises instructions to:

compare accumulated statistic values from the buckets to second threshold values to determine that an event is of significance.

63. A method of monitoring traffic flow in a monitor device disposed to receive network traffic packets comprises:

producing statistics corresponding to a parameter of traffic flow to trace the source of an attack, with producing further comprising:

mapping the traffic flow into a plurality of buckets;

varying the number of buckets according to the amount of traffic and number of flows according to down traffic flow into different buckets and examining statistics accumulated for a parameter and a corresponding threshold in the bucket.

64. The method of claim 63 wherein varying varies the number of buckets so that the monitoring device is not vulnerable to DoS attacks against its own resources.

65. The method of claim 63 wherein varying the number of buckets comprises:

comparing the number of buckets to a threshold number of buckets;

determining whether the number of buckets should be divided into more buckets or combined into fewer buckets based on comparing the number of buckets to the threshold and as the number of buckets changes, the buckets have values derived from the buckets prior to the change.

66. The method of claim 63 wherein further comprising:

comparing accumulated statistic values from the buckets to second threshold values to determine that an event is of significance.

67. The method of claim 63 wherein comparing statistic values comprises:

accumulating statistic values from the packets; and

comparing the values accumulated in the buckets to thresholds that depend on the number of buckets.

68. The method of claim 63 wherein the variable number of buckets dynamically adjusts the amount of traffic and number of flows monitored, so that the monitoring device is not vulnerable to a denial of service attack against its own resources.

69. The method of claim 63 wherein the buckets are storage areas in a memory space of the monitor device and mapping the traffic flow into a plurality of buckets comprises:

applying a hash function "f(h)" to the parameter of the traffic flow to output an integer corresponding to one of the buckets.

70. A computer program product residing on a computer readable medium for monitoring traffic flow in a monitor device disposed to receive network traffic packets comprises instructions for causing the device to:

produce statistics corresponding to a parameter of traffic flow to trace the source of an attack, with producing further comprising:

map the traffic flow into a plurality of buckets;

vary the number of buckets according to the amount of traffic and number of flows according to down traffic flow into different buckets and examining statistics accumulated for a parameter and a corresponding threshold in the bucket.

71. The computer program product of claim 70 wherein instructions to vary, vary the number of buckets so that the monitoring device is not vulnerable to DoS attacks against its own resources.

72. The computer program product of claim 70 wherein instructions to vary comprises instructions to:

compare the number of buckets to a threshold number of buckets;

determine whether the number of buckets should be divided into more buckets or combined into fewer buckets based on comparing the number of buckets to the threshold and as the number of buckets changes, the buckets have values derived from the buckets prior to the change.

73. The computer program product of claim 70 further comprising instructions to:

compare accumulated statistic values from the buckets to second threshold values to determine that an event is of significance.

74. The computer program product of claim 70 wherein instructions to compare statistic values comprises instructions to:

accumulate statistic values from the packets; and

compare the values accumulated in the buckets to thresholds that depend on the number of buckets.

75. The computer program product of claim 70 wherein the variable number of buckets dynamically adjusts the amount of traffic and number of flows monitored, so that the monitoring device is not vulnerable to a denial of service attack against its own resources.

76. The computer program product of claim 70 wherein the buckets are storage areas in a memory space of the monitor device and instructions to map the traffic flow into a plurality of buckets comprises instructions to:

apply a hash function "f(h)" to the parameter of the traffic flow to output an integer corresponding to one of the buckets.

77. The data collector of claim 21 further comprising:

a port to link the data collector to a central control center.

Evidence Appendix

None

Related Proceedings Appendix

None